REPORT DOCUMENTATION PAGE

Form Approved
OMB NO. 0704-0188

1. AGENCY USE ONLY (Leave Blank)	2. REPORT DATE 21 OCT 05	3. REPORT TYPE AND DATES COVERED FINAL REPORT 15 SEPT 03 THRU 14 JUN 05
4. TITLE AND SUBTITLE EXTERNAL MEMORY ALGORITHMS: DEALING WITH MASSIVE DATA		5. FUNDING NUMBERS DAAD19-03-1-0321
6. AUTHOR(S) JEFFREY S. VITTER		
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) PURDUE UNIVERSITY WEST LAFAYETTE IN 47907		8. PERFORMING ORGANIZATION REPORT NUMBER 531 1398-0361
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) U. S. Army Research Office P.O. Box 12211 Research Triangle Park, NC 27709-2211		10. SPONSORING / MONITORING AGENCY REPORT NUMBER 45451.9-MA

11. SUPPLEMENTARY NOTES

The views, opinions and/or findings contained in this report are those of the author(s) and should not be construed as an official Department of the Army position, policy or decision, unless so designated by other documentation.

12 a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution unlimited.	12 b. DISTRIBUTION CODE

13. ABSTRACT (Maximum 200 words)

The bottleneck in many applications that process massive amounts of data is the I/O communication between internal memory and external memory. The bottleneck is accentuated as processors get faster and parallel processors are used. The goal of this proposal is to deepen our understanding of the limits of I/O systems and massive data storage systems and to construct algorithms that are provably efficient. The three measures of performance are number of I/Os, disk storage space, and CPU time. Even when the data fit entirely in memory, communication can still be the bottleneck, and the related issues of caching become important.

14. SUBJECT TERMS Input/Output, I/O, algorithms, external memory	15. NUMBER OF PAGES 7
	16. PRICE CODE

17. SECURITY CLASSIFICATION OR REPORT UNCLASSIFIED	18. SECURITY CLASSIFICATION ON THIS PAGE UNCLASSIFIED	19. SECURITY CLASSIFICATION OF ABSTRACT UNCLASSIFIED	20. LIMITATION OF ABSTRACT UL

NSN 7540-01-280-5500

Standard Form 298 (Rev.2-89)
Prescribed by ANSI Std. 239-18
298-102

Enclosure 1

Number of Papers: 9

J. S. Vitter. ``Geometric and Spatial Data Structures in External Memory,'' in Handbook on Data Structures and Applications, (edited by D. Mehta and S. Sahni) CRC Press, to appear.

M. Wang, B. Iyer, and J. S. Vitter. ``Scalable Mining for Classification Rules in Relational Databases,'' Herman Rubin Festschrift, Lecture Notes Monograph Series, 45, Institute of Mathematical Statistics, Hayward, CA, 2004.

The conference papers are reviewed and have a higher hurdle than the journals in many cases.

R. Grossi, A. Gupta, and J. S. Vitter. ``When Indexing Equals Compression: Experiments with Compressing Suffix Arrays and Applications,'' Proceedings of the 15th Annual SIAM/ACM Symposium on Discrete Algorithms (SODA ''''04), New Orleans, LA, January 2004.

T. M. Ghanem, R. Shah, M. F. Mokbel, W. G. Aref, and J. S. Vitter. ``Bulk Operations for Space-Partitioning Trees,'' Proceedings of the 20th Annual IEEE International Conference on Data Engineering, Boston, March--April 2004.

L. Foschini, R. Grossi, A. Gupta, and J. S. Vitter. ``Fast Compression with a Static Model in High-Order Entropy,''
Proceedings of the 2004 IEEE Data Compression Conference, Snowbird, UT, March 2004.

I. Ilyas, R. Shah, W. G. Aref, J. S. Vitter, and A. Elmagarmid. ``Rank-aware Query Optimization,''
Proceedings of the 2004 ACM SIGMOD International Conference on Management of Data, Paris, France, June 2004.

S. Muthukrishnan, R. Shah, and J. S. Vitter. ``Finding Deviants in Time Series Data Streams,'' Proceedings of the 16th International Conference on Scientific and Statistical Database Management, Santorini Island, Greece, June 2004.

R. Shah, P. J. Varman, and J. S. Vitter. ``Online Prefetching and Caching for Parallel Disks,'' Proceedings of the 16th Annual ACM Symposium on Parallel Algorithms and Architectures, Barcelona, Spain, June 2004.

R. Cheng, Y. Xia, S. Prabhakar, R. Shah, and J. S. Vitter. ``Efficient Indexing Methods for Probabilistic Threshold Queries over Uncertain Data,'' Proceedings of the 29th International Conference on Very Large Databases, Toronto, CA, August 2004.

L. Lim, M. Wang, and J. S. Vitter. ``CXHist: An On-line Classification-based Histogram for XML String Selectivity Estimation,'' Proceedings of the 31st International Conference on Very Large Databases, Trondheim, Norway, August-September 2005.

Honors and Awards

Scientific progress and accomplishments Problems involving massive amounts of data arise naturally in a variety of disciplines, such as spatial databases, geographic information systems, text repositories, string databases, constraint logic programming, object-oriented databases, statistics, virtual reality systems, and computer graphics. NASA's Earth Observing System project, the core part of the Earth Science Enterprise (formerly Mission to Planet Earth), produces petabytes (10^{15} bytes) of raster data per year!
A major challenge is to develop mechanisms for processing the data efficiently, or else much of it will be useless.

The bottleneck in many applications that process massive amounts of data is the I/O communication between internal memory and external memory.
The bottleneck is accentuated as
processors get faster and parallel processors are used. Parallel disk arrays are often used to increase the I/O bandwidth.
The goal of this proposal is to deepen our understanding of the limits of I/O systems and to construct external memory algorithms that are provably efficient. The three measures of performance are number of I/Os, disk storage space, and CPU time. Even when the data fit entirely in memory, communication can still be the bottleneck, and the related issues of caching become important.

Theoretical work involves development and analysis of provably efficient external memory algorithms and cache-efficient algorithms for a variety of important application areas. In [CRChandbook], we give a broad survey of the state of the art in the design and analysis of external memory algorithms and data structures.
We address several batched and on-line problems, involving text databases, prefetching and streaming data from parallel disks, and database selectivity estimation. Our experimental validation uses our TPIE programming environment.

Most RDBMSs maintain a set of histograms for estimating the selectivities of given queries. These selectivities are typically used for cost-based query optimization. While the problem of building an accurate histogram for a given attribute or attribute set has been well-studied, little attention has been given to the problem of building and tuning a set of histograms collectively for multidimensional queries in a self-managed manner based only on query feedback.

In [Lim et al], we present SASH, a Self-Adaptive Set of Histograms that addresses the problem of building and maintaining a set of histograms. SASH uses a novel two-phase method to automatically build and maintain itself using query feedback information only. In the online tuning phase, the current set of histograms is tuned in response to the estimation error of each query in an online manner. In the restructuring phase, a new and more accurate set of histograms replaces the current set of histograms. The new set of histograms (attribute sets and memory distribution) is found using information from a batch of query feedback. We present experimental results that show the effectiveness and accuracy of our approach.

The proliferation of online text, such

as on the World Wide Web and in databases, motivates the need for space-efficient text indexing methods that support fast string searching. In this scenario, consider a text T that is made up of n symbols drawn from a fixed alphabet Sigma and that is represented in n log
|Sigma| bits by encoding each symbol with log |Sigma|
bits. The goal is to support quick search queries of any string pattern P of m symbols, with T being fully scanned only once, namely, when the index is created.

Text indexing schemes published in the literature are greedy of space and require additional Omega(n log n) bits in the worst case. For example, suffix trees and suffix arrays need Omega(n) memory words of Omega(log n) bits in the standard unit cost RAM. These indexes are larger than the text itself by a factor of Omega(log_|Sigma| n), which is significant when Sigma is of constant size, such as ascii or unicode. On the other hand, they support fast searching either in O(m log |Sigma|) time or in O(m + log n) time, plus an output-sensitive cost O(occ) for listing the pattern occurrences.

In [Grossi et al], we present a novel implementation of compressed suffix arrays exhibiting new tradeoffs between search time and space occupancy for a given text (or
sequence) of n symbols over an alphabet Sigma, where each symbol is encoded by lg |Sigma| bits. We show that compressed suffix arrays use just nH_h + O(n lg lg n/ lg_|Sigma| n) bits, while retaining full text indexing functionalities, such as searching any pattern sequence of length m in O(m lg |Sigma| + polylog(n)) time. The term H_h <= lg |Sigma| denotes the hth-order empirical entropy of the text, which means that our index is nearly optimal in space apart from lower-order terms, achieving asymptotically the empirical entropy of the text (with a multiplicative constant~1). If the text is highly compressible so that H_n = o(1) and the alphabet size is small, we obtain a text index with o(m) search time that requires only o(n) bits.
We also report further results and tradeoffs on on high-order entropy-compressed text indexes.

In [Grossi et al], we report on a new and improved version of high-order entropy-compressed suffix arrays, which has theoretical performance guarantees similar to those in our earlier work, yet represents an improvement in practice.
Our experiments indicate that the resulting text index offers state-of-the-art compression. In particular, we require roughly 20% of the original text size--without requiring a separate instance of the text--and support fast and powerful searches. To our knowledge, this is the best known method in terms of space for fast searching.

We have developed extremely promising versions of a new and improved high-order entropy-compressed suffix arrays, which has theoretical performance guarantees similar to those in our earlier work, yet represents an improvement in practice. Our experiments indicate that the resulting text index offers state-of-the-art compression. In particular, we require roughly 20% of the original text size--without requiring a separate instance of the text--and support fast and powerful searches. To our knowledge, this is the best known method in terms of space for fast searching.

We have explored the use of this structure for text compression only, without use of the indexing capabilities. The resulting method compares favorably with the best methods known.

We have also investigated algorithms for massive memory problems, such as database optimization of ranking algorithms, especially those that are "rank-aware" and that adapt dynamically to the characteristics of the data encountered. Another problem of interest deals with deviant detection in data streaming applications, in which the data comes so fast that it must be handled in a real-time manner without later processing. We have also investigated novel caching and buffering mechanisms for optimizing parallel disks.

Query optimization in IBM's System RX, the first truly hybrid relational XML data management system, requires accurate selectivity estimation of path-value pairs, i.e., the number of nodes in the XML tree reachable by a given path with the given text value. Previous techniques have been inadequate, because they have focused mainly on the tag-labeled paths (tree structure) of the XML data. For most real XML data, the number of distinct string values at the leaf nodes is orders of magnitude larger than the set of distinct rooted tag paths. Hence, the real challenge lies in accurate selectivity estimation of the string predicates on the leaf values reachable via a given path.

In [Lim et al], we present CXHist, a novel workload-aware histogram technique that provides accurate selectivity estimation on a broad class of XML string-based queries. CXHist builds a histogram in an on-line manner by grouping queries into buckets using their true selectivity obtained from query feedback. The set of queries associated with each bucket is summarized into feature distributions. These feature distributions mimic a Bayesian classifier that is used to route a query to its associated bucket during selectivity estimation. We show how CXHist can be used for two general types of (path,string) queries: exact match queries and substring match queries. Experiments using a prototype show that CXHist provides accurate selectivity estimation for both exact match queries and substring match queries.

Number of Graduate Students Supported: 1

Names of Graduate Students: Ankur Gupta

Number of Post Doctorates Supported: 2

Names of Post Doctorates: Rahul Shah and Wing-Kai Hon (partial)

Number of Faculty Supported: 1

Names of Faculty: Jeffrey Vitter

Number of PHDs Awarded: in progress